Chapter 5

Différentes Stratégies de Segmentation Basées sur les *Level-Sets* et le *Fast-Marching*

Résumé — En dépit de beaucoup d'avantages, les *Level-Sets* souffrent de sérieux inconvénients quand on les comparent à des représentations explicites ou paramétrées des surfaces actives. Dans ce chapitre, nous allons essayer de fournir une réponse à plusieurs de ces problèmes.
En ce qui concerne le manque d'interactivité, nous montrons comment implémenter quelques techniques d'interaction avec la surface implicite en section 5.1.
Par la suite nous présentons en section 5.2 une modification des forces de "région" sur lesquelles nous nous appuierons dans les applications.
Le coût de la segmentation d'une surface peut parfois être exhorbitant, lorsqu'on rajoute une dimension au problème comme c'est le cas avec les *Level-Sets* . Nous allons utiliser le *Fast-Marching* pour fournir une initialisation rapide et précise pour que le modèle plus complexe des *Level-Sets* n'ait besoin que de quelques itérations pour converger vers une solution encore plus précise. Dans les sections 5.3 et 5.4, nous présentons un algorithme de segmentation basé sur le *Fast-Marching* et la mise en oeuvre d'un algorithme combinant nos deux méthodes en une seule.

Abstract — Despite its advantages, level-set modelization suffers from several drawbacks compared with explicit and parametric models.
No local deformations are implemented, and in section 5.1, we introduce several techniques to include interactivity in the traditional *Level-Sets* framework. We also derive variations of the region-based forces, which will be much useful for our segmentation applications.
A large number of computations is often needed to solve the variational equations involved in the *Level-Sets* model. Using the *Fast-Marching* as a segmentation tool, we are going to initialize *Level-Sets* with a pre-segmentation near the solution, where only a few iterations are needed to converge to a sub-pixel accurate solution. In section 5.3 and 5.4, we present the collaboration of *Fast-Marching* and *Level-Sets* in a single segmentation framework.

5.1 Interactive Segmentation with the Level-Sets Framework

In medical imaging no automatic method is known to be generally applicable, completely reliable and robust. As a consequence, interaction is a part of many segmentation procedure to be combined to the automated segmentation process. No interactions were introduced in level-sets models, and the effect of the parameters on the model is often non-intuitive, as shown in figure 5.1.

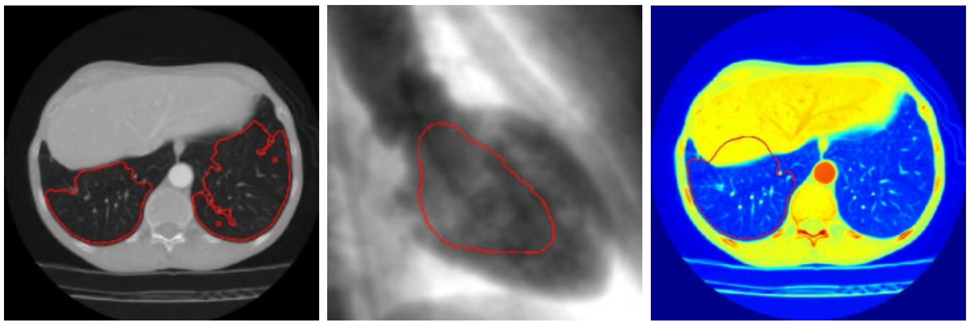

Figure 5.1. Failed segmentation with Level-Sets paradigm - Left image: The gradient influence is too important, and the level-sets is attracted by any spurious edges (vessels and arteries in the lungs); middle image: the curvature forces being too important, the inflation force is unable to reach the edges of the left ventricle; on the lung image the edges are now not sufficiently valued in order to stop inflation into the liver which surrounds the lung.

5.1.1 Interactivity requirements for the level-sets paradigm

As emphasized by most of the comparative study of explicit and implicit deformable models, and particularly in the PhD thesis of *J. Montagnat* [120], the formalism of the *Level-Sets* implicit representation restricts severely user interactivity.

The interactive requirements, when an automatic method fails, can be classified in different categories, like

- unseen evidence: the user perceive an edge where the automatic method does not (when the object intensity does not fit). In this case, the method should offer a regionally different edge defining operation or a regional parameter tuning (all examples in figure 5.1 are cases of unseen evidence);
- absence of evidence: neither the user nor the automatic method observe an edge. Interaction should imply a nullification of the evidence from the data in the region;
- dislocated evidence: the automatic method has found proof of an object according to the object model, but has confused the object with a neighboring object.

Interaction should confine the admissible contours to the indicated zone.

The definition of the correct interactivity framework is more than a difficult task, maybe more than the settle of an automatic method for medical imaging problems. The tremendous work of *S. Olabarriaga* [133, 134] on the subject settled the basis for a general interactive segmentation framework, but solutions surely lie in dedicated approaches to specific problems.

Simple interactions have been implemented in 2D for tests. They are based on modifications of the data or the model, which are two cases to answer the problems of unseen evidence and absence of evidence.

5.1.2 Fixing a point of the contour

This force will attract the zero level-set of ϕ to a user defined point. If we apply the classical evolution equation (4.5) of [135], with an added external force to the point $\mathbf{x_0}$:

$$\phi_t \quad + \quad \tilde{F}|\nabla\phi| = \phi_t + F|\nabla\phi| + \mathcal{F}_{\mathbf{x_0}}|\nabla\phi| = 0 \tag{5.1}$$

with the external force compute at each pixel $\mathbf{x}$ by

$$\mathcal{F}_{\mathbf{x_0}}(\mathbf{x}) = d(\mathbf{x_0};\mathbf{x}) \tag{5.2}$$

This force is applied to the nearest voxels of $\mathbf{x_0}$ (in a user-chosen neighborhood), on the zero level-set. In order to compute $\mathcal{F}$ for an undefined number of fixed point $\{\mathbf{x_0},\dots\mathbf{x_N}\}$ in equation (5.2), we use the *Fast-Marching* algorithm, detailed in chapter 1, as explained in table 5.1. Figure 5.2 represents iterations of the evolution

Algorithm for Computing Interactivity Force $\mathcal{F}$
at iteration i, let $\{\mathbf{x_0},\dots,\mathbf{x_N}\}$ be the N user defined fixed points, and N action maps $U_0,\dots,U_N$. For $j \in [1;N]$:

- $U_j(\mathbf{x_j}) = 0$, and $U_j(\mathbf{x}) = \infty$ elsewhere;
- propagate a front by computing action map U_j with $\|\nabla U_j\| = 1$ using *Eikonal equation* (1.6);
- stop when visiting a pixel $\mathbf{y}$ where ϕ sign changes.

The additional force $\mathcal{F}$ is given by $\mathcal{F}(\mathbf{x}) = \min_{j\in[1;N]} U_j(\mathbf{x})$ if $\mathbf{x}$ has been visited, in other case $\mathcal{F}(\mathbf{x}) = 0$.

Table 5.1. *Fast-Marching* for Computing external forces

process with *On-The-Fly* user interaction of a *Level-Sets* initialized by the distance to a circle, and using as evolution equation:

$$\phi_t + \mathcal{F}_{\mathbf{x_0}}|\nabla\phi| = 0 \tag{5.3}$$

where $F = \mathcal{F}$ in equation (5.1). Several comments can be added to this algorithm:

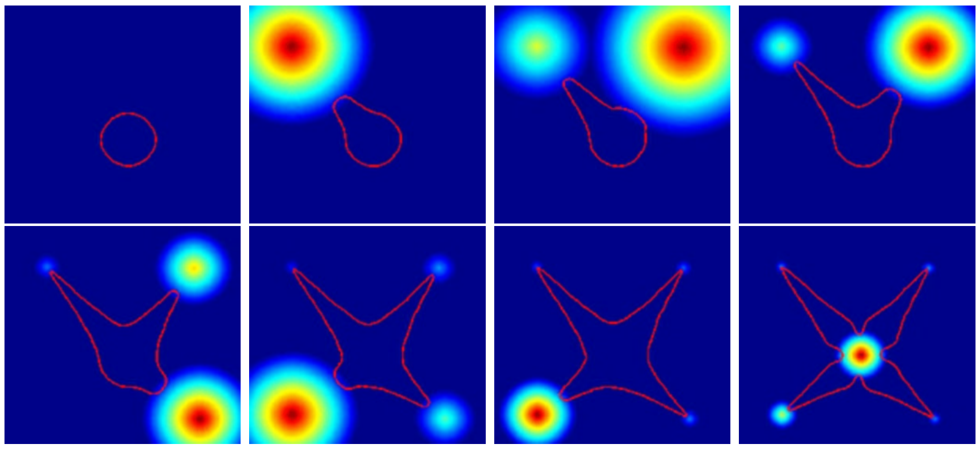

Figure 5.2. Fixing a point of the contour: these images represent the deformation of the zero level-set of ϕ, when fixing different points of the contour. The zero level-set extracted is represented in red, and is super-imposed to the external force $\mathcal{F}$ in equation 5.2

- choosing $\mathcal{F}(\mathbf{x}) = \min_{j\in[1;N]} U_j(\mathbf{x})$ instead of $\max_{j\in[1;N]}(U_j(\mathbf{x})$ in table 5.1, is just a matter of convention. The force applied is not directional, because the evolution of the curve is in its normal direction and do not integrate any tangential term. Therefore, the zero level-set is not attracted more or less by two neighboring pixels, and this is just the intensity of this force that can be tuned;
- concerning the different action maps $U_0, \ldots, U_N$, they can be integrated in only one action map U, according to the previously mentioned convention. This can greatly reduce the computing cost of N float images, for each action map. This is easily handled by taking the minimum of the different actions, because the *Fast-Marching* algorithm has been built on this minimality principle. It was the method used in figure 5.2.

5.1.3 Re-initializing the contour

The interaction we study now is a re-initialization of the contour, on the model of the *Convolution surface* developed in [14], for implicit function modeling in computer graphics (see [13]). A convolution is a modification of a signal by a filter; here the skeleton is the signal and the filter is a Gaussian kernel. More than a deformation process, this is a re-initialization of the model.

Considering our function ϕ defined on a small narrow-band, as done by *Whitaker* [184], with its *Sparse-Fields* method the level-set function is close to a binary image, as shown in figure 5.3. Defining a point $\mathbf{x_0}$ for the interaction, we compute its distance to the zero level-set, and its corresponding nearest point $\mathbf{x_1}$ on this zero level-set. In our 2D example, the segment $[\mathbf{x_0}, \mathbf{x_1}]$ is the skeleton that we convolved with a

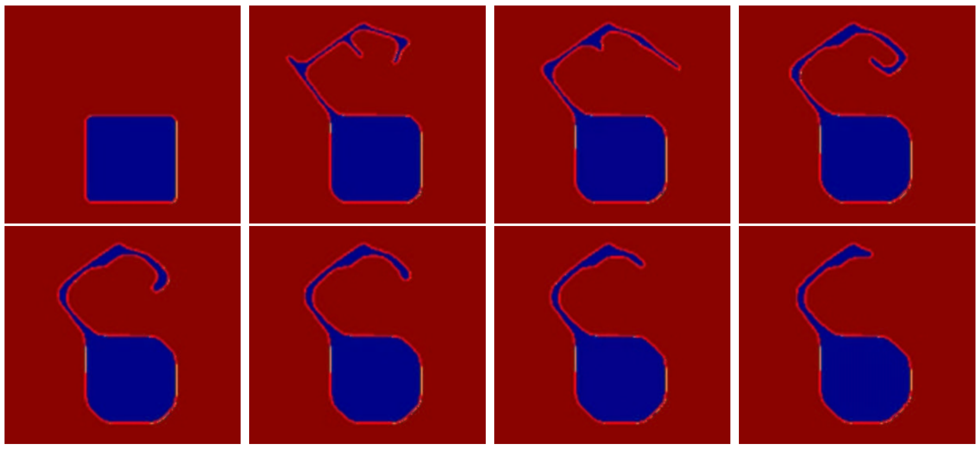

Figure 5.3. Re-initializing the contour: The initial contour which evolves according to an equation of the form $\phi_t + \kappa|\nabla\phi| = 0$ is deformed by convolution. Different steps show the curvature motion that shrinks the obtained closed curve, as explained in [70].

Gaussian kernel. We built a function $\mathcal{F}$ that for any point $\mathbf{p}$ on the image domain gives

$$\mathcal{F}(\mathbf{p}) = a\left(\int_{x_0}^{x_1} e^{-\|\mathbf{p}-\mathbf{u}\|^2/2b} du - 1/2\right) \tag{5.4}$$

that can be approximated by

$$\mathcal{F}(\mathbf{p}) = a\left(\sum_i e^{-\|\mathbf{p}-\mathbf{x_i}\|^2/2b} - 1/2\right) \tag{5.5}$$

where $\mathbf{x_i}$ are the points on the skeleton, and a, b are positive constants to control the amplitude and sharpness of the convolved object. Then we apply $\mathcal{F}$ by reconstructing the new function φ defined by the two regions:

- $\varphi^- = \phi^{-1}(\mathbb{R}^-) \cup \mathcal{F}^{-1}(\mathbb{R}^-)$
- $\varphi^+ = \phi^{-1}(\mathbb{R}^+) \cap \mathcal{F}^{-1}(\mathbb{R}^+)$

where

- if $\mathbf{x} \in \varphi^-$, $\varphi(\mathbf{x}) = \max(\phi(\mathbf{x}), \mathcal{F}(\mathbf{x}))$;
- if $\mathbf{x} \in \varphi^+$, $\varphi(\mathbf{x}) = \min(\phi(\mathbf{x}), \mathcal{F}(\mathbf{x}))$.

Applying this method to ϕ, we modify the zero level-set as done in figure 5.3. This implementation of a re-initialization is rather similar to the method proposed in [141].

5.1.4 Conclusion on the interactivity

We have shown two possible interactivity techniques, based on a modification of the data, and on a modification of the model, both based on computer graphics ideas. Basically, a lot of techniques first studied for computer graphics can be applied to the *Level-Sets* which basis is an implicit representation. The implicit representation in computer graphics (see [13]) has much in common with the formalism developed in [135]. It was even studied by *Cani and Desbrun* in [19] and detailed in [41], with a *Level-Sets* implementation for a "skin" that contains spherical particles for the simulation of highly deformable models. But in Computer Graphics, first matter is the rendering of animations, and not their computing costs. Unfortunately, interactivity requires a direct and fast output of any user interaction, which is something that is still difficult to manage with the *Level-Sets* paradigm.

But the solution probably lies in the combination of two different techniques: one for segmentation, and one for interaction. A very good example of this philosophy can be found in [188]: the authors use circular oriented particles to sample and control implicit surfaces. This technique is also used by *Szeliski et al.* [171]. The model is a force-feedback system where particles try to match the zero level of the implicit function, and where the surface follows particles. A simple constraint locks the set of particles onto a surface while the particles and the surface move. The particles use mutual repulsive forces and fissioning to sample the whole surface, and the user interactivity can be applied locally on one particle at a time, using them as control points for direct manipulation, as shown in figure 5.4[1]. Finally, very interesting work

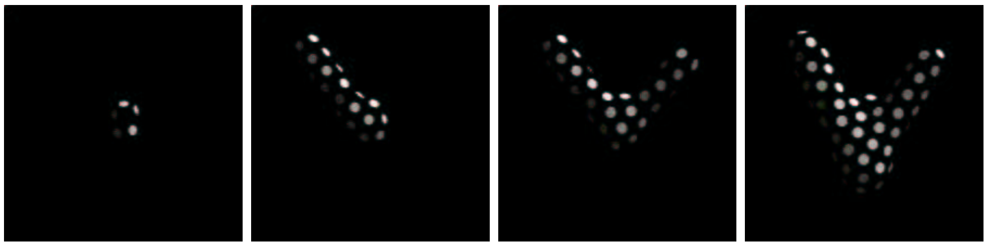

Figure 5.4. Interactivity on an implicit surface : the particles track the implicit surface, leading to an easy visualization of its zero level-set, and at the same time input a control on this surface, in order to modify its shape.

has been done on the construction of subjective contours in [154], where the *Level-Sets* extracts contours in famous test images of *Kanizsa* [81] that strongly requires image completion.

[1]We would like to acknowledge Andrew Witkin and Paul Heckbert, for providing this sequence that is an illustration of their article [188].

5.2 Region Based Forces

5.2.1 Gaussian descriptors

We recall the descriptors first introduced by *Zhu and Yuille* [196]. They are “fully automatic” in the sense that no user-defined parameter is necessary to their definition. They are based on region probabilities defined by Gaussian distributions:

$$P_{in}(x,t) = \frac{1}{\sqrt{2\,\pi}\,\sigma_{in}(t)}\ \exp(-\frac{(I(x)-\mu_{in}(t))^2}{2\,\sigma_{in}^2(t)})$$

A similar expression is used for the definition of $P_{out}(x,t)$. Here $\mu_{in}(t)$ and $\sigma_{in}^2(t)$ are respectively the mean and the variance of the image intensity over $\Omega_{in}(t)$:

$$\mu_{in}(t) = \frac{\int_{\Omega_{in}(t)} I(x)\,dx}{\int_{\Omega_{in}(t)}\,dx}\ ,\ \sigma_{in}^2(t) = \frac{\int_{\Omega_{in}(t)}(I(x)-\mu_{in}(t))^2\,dx}{\int_{\Omega_{in}(t)}\,dx}$$

This means that the histograms of the intensity in $\Omega_{in}(t)$ and $\Omega_{out}(t)$ are *modeled* by Gaussian distributions.

5.2.2 Modified Gaussian descriptors

The preceding model has one major drawback: if one of the variances is much smaller than the other one, then undesired results can be observed. For example, if the image is composed of a bright region with a small variance (a contrast-enhanced organ for instance) in the middle of a dark background with a large variance, then very bright pixels may have a greater probability to be in the background than in the bright region. This is perfectly normal from the point of view of the Gaussian model, but it is in contradiction with our *a priori* knowledge about the image informational content. In other words, the Gaussian model may be unadapted to some images.

In such cases, we can introduce an priori knowledge by modifying the two Gaussian distributions. If we know that the image region that we want $\Omega_{in}(t)$ to cover is supposed to be brighter than the one covered by $\Omega_{out}(t)$, then we take:

$$P_{in}(x,t) = \begin{cases} \frac{1}{\sqrt{2\,\pi}\,\sigma_{in}(t)}\ \exp(-\frac{(I(x)-\mu_{in}(t))^2}{2\,\sigma_{in}^2(t)}) & \text{if}\quad I(x) < \mu_{in}(t) \\ \frac{1}{\sqrt{2\,\pi}\,\sigma_{in}(t)} & \text{if}\quad I(x) \geqslant \mu_{in}(t) \end{cases}$$

and:

$$P_{out}(x,t) = \begin{cases} \frac{1}{\sqrt{2\,\pi}\,\sigma_{out}(t)} & \text{if}\quad I(x) < \mu_{out}(t) \\ \frac{1}{\sqrt{2\,\pi}\,\sigma_{out}(t)}\ \exp(-\frac{(I(x)-\mu_{out}(t))^2}{2\,\sigma_{out}^2(t)}) & \text{if}\quad \mathcal{I}(x) \geqslant \mu_{out}(t) \end{cases}$$

It is also possible to manually choose the means and variances instead of computing them from the region histograms, which can be very useful in some applications (they can be obtained through an initialization process). In this case, the region descriptors become time-independent.

5.2.3 Sigmoid descriptors

For the images where the modified Gaussian model failed, we used user-defined time-independent sigmoid probabilities. We took:

$$P_{in}(x,t) = \frac{1}{1 + e^{a_{in}\,(b_{in} - I(x))}} \tag{5.6}$$

with a similar expression for $P_{out}(x,t)$. While Gaussian have quadratic logarithms, sigmoid functions have asymptotic linear logarithms. For example, if a_{in} is positive, then:

$$\lim_{I(x)\to+\infty} \log P_{in}(x,t) = 0$$

and:

$$\log P_{in}(x,t) \sim_{I(x)\to-\infty} a_{in}\, I(x).$$

Those descriptors for the aorta of figure 3.12 are shown in figure 5.5.

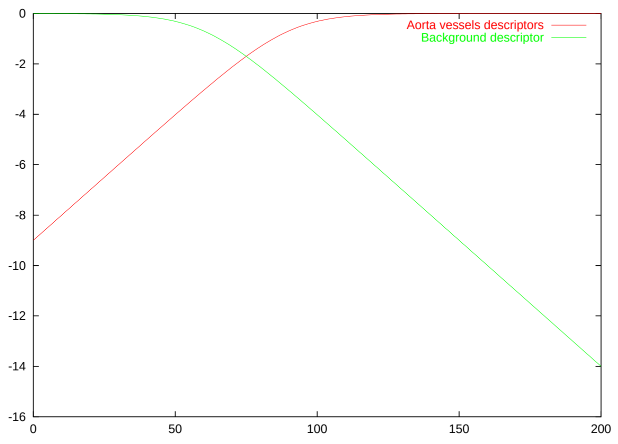

Figure 5.5. Sigmoidal region-based forces: they are used for discriminating the aorta towards the background in figure 3.12.

Figure 5.6 shows iterations of the segmentation of the aorta of figure 3.12 with sigmoidal region descriptors of figure 5.5.

5.2.4 Numerical implementation of the level-sets paradigm

Adaptive time step

We used an Euler first-order explicit time scheme for solving (4.6). We recall that explicit time schemes often require a limitation of the time step to be stable. Here it is possible to control the Courant-Friedrichs-Lewy (or CFL) number[2] of the discretized flow equation, by adapting the time step between each time iteration.

[2]Number of cells crossed by a characteristic curve during a time step.

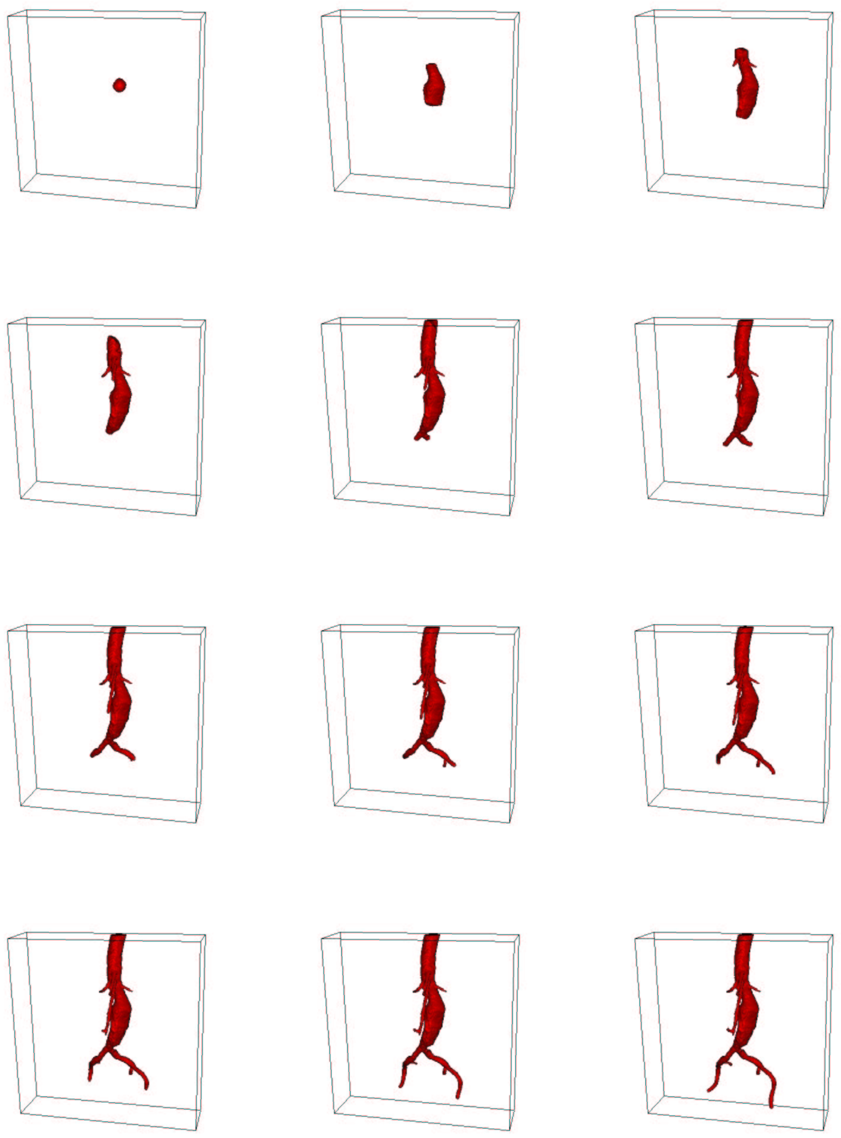

Figure 5.6. Extracting the aorta with region-based forces: the initialization is a sphere located inside the aorta object. Using the *a priori* distribution of the grey levels inside the aorta and boundary based forces, we segment the aorta, as shown on the several iterations.

The only difficulty is related to the scheme associated with the curvature flow, which has no obvious stability conditions. We have empirically chosen to give a stronger penalty to its contribution to the global CFL number. Explicitly in 3D, for a composite flow of the following form:

$$\mathbf{V} = (\beta(\mathbf{x},t) - \varepsilon(\mathbf{x},t)\,\kappa_M(\mathbf{x},t))\mathbf{n} + \mathbf{U}(\mathbf{x},t)$$

we ensure that:

$$\max_{(i,j,k)} (\nu^{\beta}_{ijk} + 3\,\nu^{\varepsilon}_{ijk} + \nu^{U}_{ijk}) \leqslant 1$$

where:

$$\begin{aligned}
\nu^{\beta}_{ijk} &= \Delta t\,|\beta_{ijk}|\sqrt{\Delta x^{-2} + \Delta y^{-2} + \Delta z^{-2}} \\
\nu^{\varepsilon}_{ijk} &= \Delta t\,|\varepsilon_{ijk}|\sqrt{\Delta x^{-2} + \Delta y^{-2} + \Delta z^{-2}} \\
\nu^{U}_{ijk} &= \Delta t\left(\frac{|U_{x\,ijk}|}{\Delta x} + \frac{|U_{y\,ijk}|}{\Delta y} + \frac{|U_{z\,ijk}|}{\Delta z}\right).
\end{aligned}$$

The results are very satisfying, and no numerical instability has been noticed during the segmentation tests.

Narrow Banding

For reasons of causality, it is possible to restrain the computation domain to a band of cells around the zero-level set of $\phi(\cdot,t)$. The result is a decrease of the computational cost. Several approaches have already been proposed, but we have build a new one which is adapted to our segmentation problems.

Classical approaches are referred to as narrow-band methods (see figure 5.7). Some are based on combinatory studies in order to add or remove cells around $\phi(0,t)$ as it moves (see *Adalsteinsson and Sethian* [2]). Another approach can be found in the thesis of *Keriven* [84], and consists in computing the signed distance function to $\phi(0,t)$ when it gets too close to the boundaries of the band, and re-initializing $\phi(\cdot,t)$ with the computed distance function. Other authors chose to initialize ϕ with a distance function and maintain $\|\nabla\phi\| = 1$ at each iteration (see *Adalsteinsson and Sethian* [3], and *Gomes and Faugeras* [68]).

These methods cited above are efficient, especially when the initial condition $\phi(\cdot,0)$ is far from the solution. Here we propose a new specific method that is very efficient for the segmentation of fine- to medium-size tubular structures (like arterial, venous, or bronchial trees), or when using a rough pre-segmentation for initializing ϕ.

The idea is simple: we start from a distance function, either associated to a pre-defined geometric primitive, or computed from a binary pre-segmentation by a fast-marching algorithm (see next chapter). We define a narrow-band area and a wide-band area around $\phi(0,0) = \phi_0$ (the wide-band is larger than narrow-band and comprises it). The calculations are then performed in every cell of the wide-band area. If $\phi(0,t)$ gets out of the narrow-band area (i.e. the sign of ϕ changes somewhere

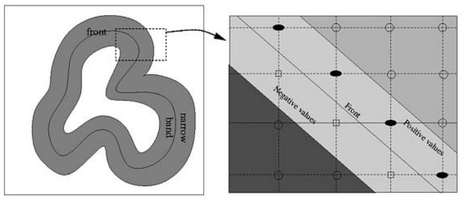

Figure 5.7. Narrow Band Diagram: Left image shows the overview of the narrow-band structure, and right image shows the detail.

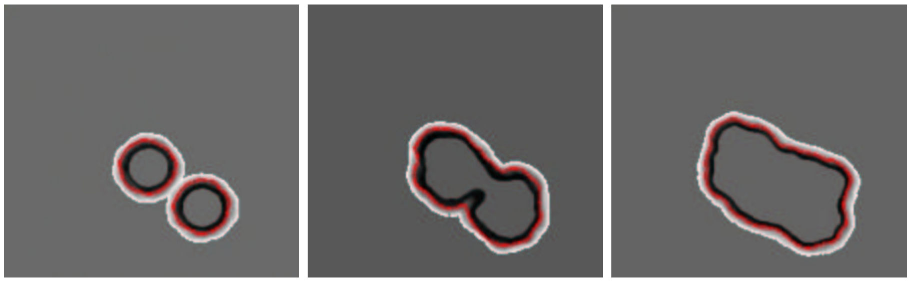

Figure 5.8. Narrow Band reconstruction: images are samples of the evolution and reconstruction of the narrow-band, starting from two circles that merge.

in the outside of the narrow-band area), then both bands are locally enlarged in a spherical manner.

As a consequence, we can add cells to the computational domain, but we never remove cells from it. If a pre-segmentation is used, the enlargements of the bands are few and the resulting computational cost is very small. This method has also proved to be efficient in the case of propagation in tubular structures with diameters comparable to the wide-band width (for example 10 times a cell diagonal).

Boundary conditions

We used so-called *free* boundary conditions, that is to say that we extrapolate the values of $\phi(\cdot, t)$ in the outside vicinity of the computation domain boundaries by taking the closest value inside the domain, and that the corresponding finite differences are equal to zero. Some authors use other entropic boundary conditions. The main idea is that the information that goes out of the computation domain is lost, and that no information should be introduced into the domain through its boundaries.

5.3 Using the Fast-Marching for segmentation

Originally introduced by *Malladi and Sethian* [111], the *Fast-Marching* can be used as a fast initialization algorithm for image segmentation. Despite the fact that it requires low computational cost, it has several drawbacks, in particular that the speed F in equation (4.20) is always positive or always negative. Therefore, the front evolves and propagates in the whole image domain, without being stop by any edge, and it cannot be used for cases with curvature-dependent speed functions. The stopping criterion still relies on a user perception of the segmentation, like visual inspection of the domain visited by the algorithm during propagation, or the choice for an appropriate stopping time, as done in [111]. But the nice segmentation properties of the *Fast-Marching* algorithm shall be taken into account, as shown in figure 4.10, where we segment a brain on the basis of the method described in [111].

Still, the stopping criterion in this case was to visually control the segmentation process, because the automatic detection of the crossing of the interesting edges is difficult to implement.

Noticing that *Eikonal equation* is used to model the continuous formulation of the watershed transform, we have developed an interactive segmentation tool based on the flooding process of this morphological algorithm introduced in [180]. It is based on the same principle that segmentation boundaries are defined by pixels where several evolving fronts collide.

5.3.1 Comparison with the watershed algorithm

The classical morphological segmentation paradigm [156] is based on the watershed transform [181]. The method is very simple:

- the gradient image $\|\nabla I\|$ is computed (and enhanced);
- for each object of interest, an inside particle is detected (either interactively or automatically);
- flooding waves are propagated from the set of markers and flood the topographic surface $\|\nabla I\|$.

The points where the flooding waves meet each other form the segmentation boundaries. Different regions between boundaries are called catchment basins. Usually markers are the regional minima of the gradient image. An example of the watershed transform is shown in figure 5.9. Very often, the minima are extremely numerous, leading to an over-segmentation, therefore for practical cases, the watershed transform will take as seed points a smaller set of markers, identified through pre-processing techniques.

The flooding waves are propagated according to the topographic map $\|\nabla I\|$, where the pixels belonging to a catchment basin are nearer to its corresponding regional minimum than to any other minima. In this framework, the construction of catchment basins can be seen as a shortest path problem between a marker and the image points. It corresponds to compute the gray-weighted distance transform (GWDT) of the image to the set of markers. There are several ways to compute this $GWDT$:

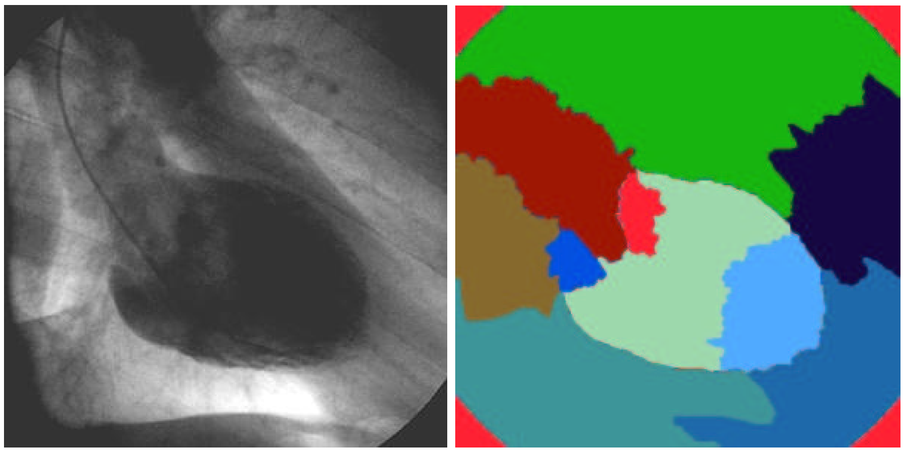

Figure 5.9. Watershed transform of a 2D X Ray image of the heart (LV): The seed points are regional minima of the gradient of image 5.9-left; the flooding was implemented in the continuous framework of *Eikonal equation* , and the regional minima were filtered in order to have a smaller set of markers.

as shown in [118], viewing the topographic image domain as a refractive index, as explained in chapter 1, the distance between a marker and any point in the image is the line integral of the penalty $\|\nabla I\|$ along the optical path length. We reformulate *Eikonal equation* 4.22 with

$$\|\nabla T(\mathbf{x})\| = \|\nabla I(\mathbf{x})\| \tag{5.7}$$

and the *GWDT* can be computed on the whole image domain, using the markers as sources pixels p where $T(\mathbf{p}) = 0$.

Maragos has shown in [118] that the continuous segmentation approach based on the *Eikonal equation* outperforms the discrete segmentation results. However several problems remain important:

- the final segmentation relies on the number of the markers: too many markers lead to an over segmentation of the image, and merging algorithms are needed as post-processing;
- it relies also on the quality of the edge detector applied to the real image: if there is a gap in the edge information, the nearest marker will flood the adjacent region.

We have devised a very simple method based on user interactivity, and different front speeds, based on the formulation of the flooding of the whole image. Each front, initialized by a user-defined seed point, will propagate according to its own propagation speed, derived from local image information (and not just gradients), and will stop when it meet other propagating fronts. This method has been implemented to provide a *quick and dirty* initialization for the *Level-Sets* method.

5.3.2 Interactive segmentation with the Fast-Marching

Flooding algorithm

In the following, we assume that we have an undefined number of seed points for front propagation, that can be labeled with a known number of labels. We detail below the flooding algorithm for multi-label front propagation.

Definition

- a set of starting point $\mathbf{p}_1, \dots, \mathbf{p}_n$, located inside the image domain;
- each starting point $\mathbf{p}_i$ is assigned a label l_i $i \in [1; n]$ (not necessarily different);
- a label map $\mathcal{L}$, for controlling collision;
- a penalty image $\mathcal{P}$ which will drive the front propagation, and which is a function of the position and the label of the front;
- We initialize the usual set of data-structures for front propagation, including an action map $\mathcal{A}$ and (only) one min-heap structure $\mathcal{H}$;

Initialization

- we initialize a classical front propagation method, setting $\mathcal{A}(p_i) = 0$ $\forall i \in [1; n]$ and storing all seed points in the min-heap structure;
- $\mathcal{L}(\mathbf{p}_i) = l_i$, $\mathcal{L} = -1$ elsewhere;

Loop: at any iteration

- Let $\mathbf{x}_{min}$ be the *Trial* point with the smallest action $\mathcal{A}$;
- Move it from the *Trial* to the *Alive* set (i.e. $\mathcal{A}_{\mathbf{x}_{min}}$ is frozen);
- Update $\mathcal{P}(\mathcal{L}(\mathbf{x}_{min}))$
- For each neighbor $\mathbf{x}$ (6-connexity in 3D) of $\mathbf{x}_{min}$:
  - If $\mathbf{x}$ is *Far*
    1. add it to the *Trial* set;
    2. $\mathcal{L}(\mathbf{x}) = \mathcal{L}(\mathbf{x}_{min})$
    3. compute $\mathcal{A}$ according to equation

       $$\|\nabla \mathcal{A}\| = \mathcal{P}\left(\mathcal{L}(\mathbf{x}_{min})\right) \tag{5.8}$$

       using the up-wind discretization scheme, involving only the neighbors of $\mathbf{x}$ with label $\mathcal{L}(\mathbf{x}_{min})$;
  - If $(\mathbf{x})$ is *Trial*
    1. recompute the action $\mathcal{A}(\mathbf{x})$ with equation(5.8), involving only the neighbors of $\mathbf{x}$ with label $\mathcal{L}(\mathbf{x}_{min})$;
    2. if the new value is smaller, then $\mathcal{L}(\mathbf{x}) = \mathcal{L}(\mathbf{x}_{min})$

Termination

- stop when all pixels on the image domain are visited;
- the label map $\mathcal{L}$ represents the final segmentation into k regions R_k.

In this algorithm, we have deliberately left blank the update procedure of the potential. Using a very simple potential based on the grey level information. Having the n seed points $\mathbf{p}_1, \ldots, \mathbf{p}_n$, we can set at initialization

$$\text{if } \mathcal{L}(\mathbf{x}) = \mathcal{L}(\mathbf{p}_i) \ , \ \mathcal{P}(\mathbf{x}) = \max(I(\mathbf{x}) - I(\mathbf{p}_i), 0) + w \tag{5.9}$$

Then the speed is inversely proportional to the difference between the grey-level of the starting seed point and other points in the image. Result of this strategy can be observed in figure 5.10 where different seed points were manually located on the image 5.10-left, leading to the segmentation of figure 5.10-right. The resulting is not

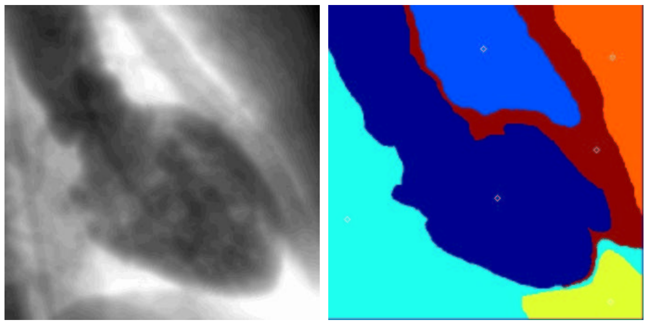

Figure 5.10. Front competition on a 2D X Ray image of the heart (LV): The seed points were manually located on the dataset on the left image; they are visible as white dots on the right image.

convincing, but the use of this kind of information to differentiate the different front speeds has a bigger potential than the classical watersheds formulation. If there is a gap in the gradients between two basins, one of the two fronts could flood completely into the adjacent basin with the watersheds, whereas the speed in the adjacent region could still be discriminating the unwanted flooding, in the case of our competitive fronts algorithm. This is mainly the difference between a difference of a *plateau* and a barrier in the penalty (where *plateau* is a region with small variations in the intensity).

Thus, different strategies can be implemented, as many as there are different possible potentials.

Setting the penalty information

We can devise a very simple algorithm that do not use the information coming from the seed points, but for the m first visited voxels for each label. This reduces greatly the influence of the user interaction in setting the markers position. Therefore for each label,

- At iteration i in the *Fast-Marching* we examine the point $\mathbf{x}$ in the min-heap whose action is minimal;
- We remove it from the heap and we examine its label $l = \mathcal{L}(\mathbf{x})$ and its grey level $I(\mathbf{x})$;

- For the front points with label l, we know N_l the number of points considered and μ_l the mean grey level value of those points;
- if $N_l < m$, we update this mean grey level value for the label l

$$\mu_l = (N_l * \mu_l + I(\mathbf{x}))/(N_l + 1) \ , \ N_l + + \tag{5.10}$$

- considering that the image grey level of the pixel in the desired region are a random distribution of mean μ_l we can also update the variance with *Koenig-Huyghens* formula $\sigma_l^2[X] = \mu_l[X^2] - \mu_l[X]^2$ for the random variable X;
- we use this new mean values μ_l and σ_l in the computations of the action at each neighbors of $\mathbf{x}$.

In figure 5.11 is shown an example of using this strategy for setting the penalty term. The speed function in the image shown is computed with $m = 10$ for all

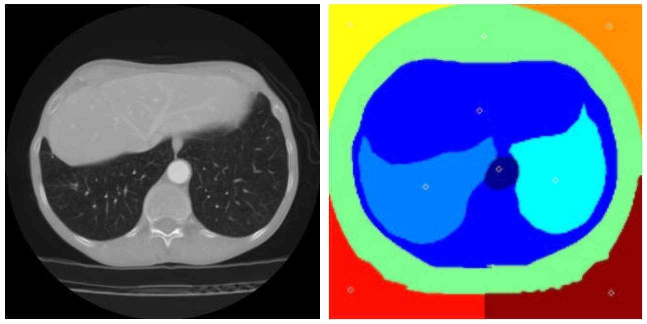

Figure 5.11. Front competition on a 2D scanner slice of the lungs: The seed points were manually located on the dataset on the left image; they are visible as white dots on the right image.

labels. The speed function for each label l is computed with the penalty $\mathcal{P}(\mathbf{x}) = e^{-|I(\mathbf{x})-\mu_l|/2\sigma_l^2}$.

Unfortunately, this algorithmic trick makes the penalty a time-dependent function, and thus the minimality principle is violated, and *Eikonal equation* discretization cannot be applied in theory during the iterations where $\mathcal{P} = \mathcal{P}(\mathbf{x}, t)$. Therefore, this heuristic must be seen as a region growing algorithm to operate a statistical study at the beginning of the segmentation process before visiting the whole image domain. It can also be replaced by a similar statistical study in a sphere around the seed points.

Figure 5.12 displays the result of the same strategy on the brain dataset used for test in figure 4.10. This result was obtained by setting different seed points in the different objects in the brain MR image which is represented on the left column of figure 5.12. Each one of the regions was initialized with one marker, and they are represented when all fronts collide super-imposed on three orthogonal views of the dataset in the left column of figure 5.12. Inconvenient of this kind of segmentation is that it assumed that the correct number of classes is known *a priori* and that the user can clearly indicate to the algorithm the location of each connected components

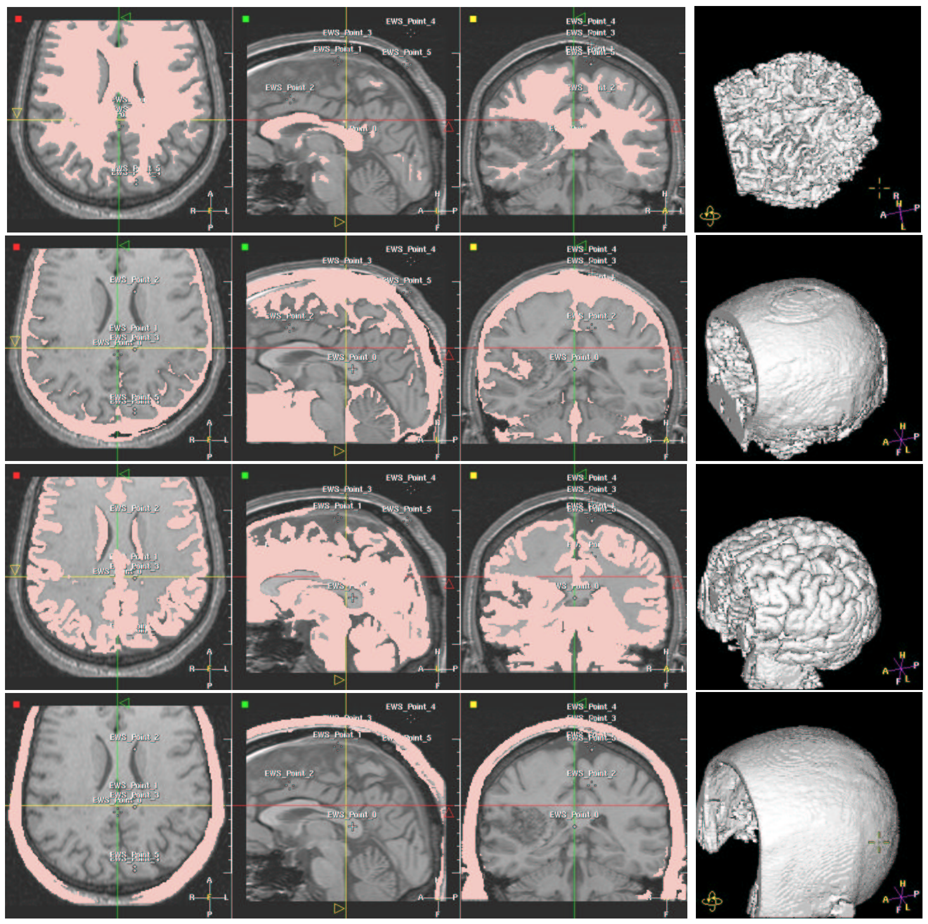

Figure 5.12. Front competition result on the brain: Simultaneous segmentation of the white and grey matter of the brain, plus the skull and the skin; each row corresponds to a region, and the intersection between the regions and the datasets is represented in pink, in the left column.

of this class. But target here is to provide a quick and dirty initialization to a more complex and time consuming method (like *Level-Sets*).

The same test was done on a lung image, where we try to separate the airways from the soft tissue and vessels, in figure 5.13. Results The underlying dataset is a volume-of-interest extracted from a multi-slice CT scanner of the lungs. The dataset, almost isotropic, contains a huge number of pixels, and even a fast algorithm like ours is not able to achieve this result in “interactive time”. But we have shown the ability of this simple algorithm to achieve difficult segmentation tasks.

Further, in this algorithm, there is no need to use several data-structures (like min-heap) to store each different front. Due to the minimality principle of the *Fast-*

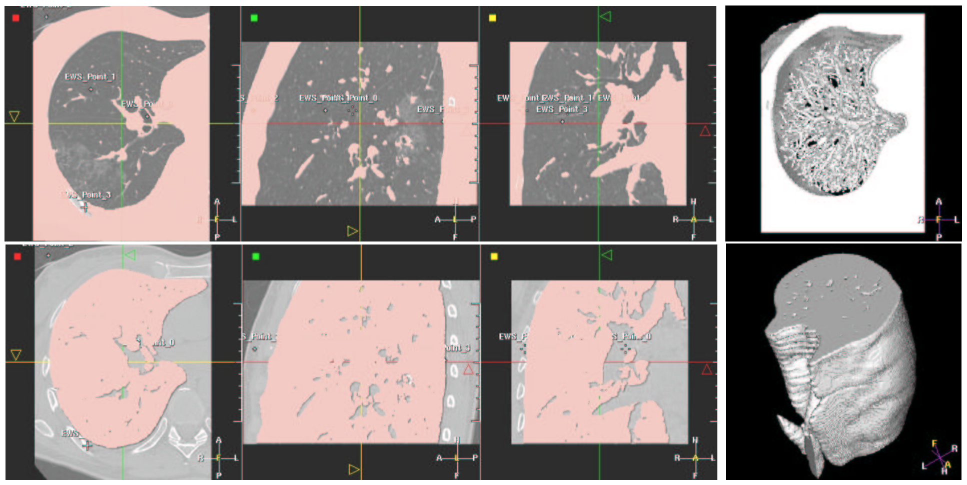

Figure 5.13. Front competition result on the lungs: separating air and soft tissues in a multi-slice CT dataset of the Lungs; seed points are manually located in the different parts of the image (on in the airways, on in the tissues, and on outside the object).

Marching construction, the point with minimum energy, independently from its label, will still be at the root of the min-heap data-structure. And each point of the image will be visited only one time, leading to approximately the same computing cost than the propagation of a single front in the image domain. It seems that a similar version of the multiple front propagation algorithm has been developed in [103, 167], for initializing color and texture segmentation with the *Level-Sets* methods.

5.4 Combining Fast-Marching and Level Sets

5.4.1 Advantages and Drawbacks of *Fast-Marching*

The *Fast-Marching* algorithm is used for the computation of a minimal action map, or weighted distance transform (from the morphological point of view).

Drawbacks : monotonicity of the speed (can only propagate in one direction), no control of the curvature of the contour to be extracted, no stopping criterion (speed always strictly positive/negative).

Advantages

- It is a very fast algorithm (as shown in the case of path extraction for virtual endoscopy in chapter 3);
- it can segment objects with complex topologies (see the brain example in figure 4.10);

- it can be used with simple initialization of seed points (see the competitive front segmentation of figure 5.12);
- There are lots of Penalty definition (depending on user's imagination, ranging from grey-levels to Hessian measures [60], and other complicated measured obtained through filtering).
- it can be used for initialization: Figure 5.14 shows images of the conversion from an implicit representation of the object, to an explicit model.

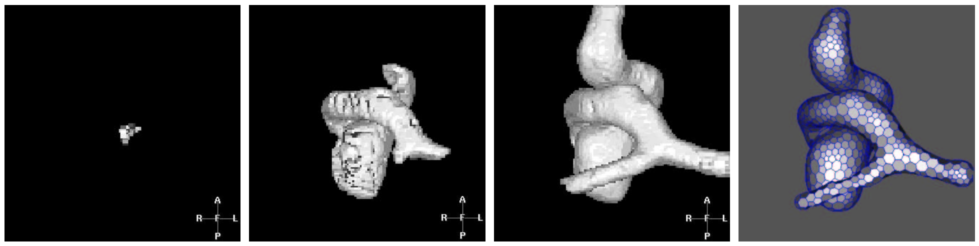

Figure 5.14. Combining Fast-Marching with Explicit Methods : the first three images are samples of the segmentation process described in section 5.3; the right image is the conversion of the implicitly defined model into a simplex mesh.

Drawbacks

- Monotonicity of the speed (can only propagate in one direction);
- No Curvature term in the Energy of the contour to be extracted;
- No stopping criterion (Speed always strictly positive/negative).

5.4.2 Advantages and Drawbacks of *Level-Sets*

Level-Sets implementation allows to evolve embedded curves with wide variety of force models, and lead to high-accuracy segmentations with sub-pixel precision. But they often use time-consuming iterative approaches. They have always been criticized because of their computational cost. For instance being used for the segmentation of the cortex (see for example *Zeng et al.* [195]), they provide very satisfying results, but several hours of computation are needed. Several implementations try to override this problem.

The speed of the algorithm has been increased with the use of semi-implicit discretization schemes by *Goldenberg et al.* [65, 67]. This method is based on the implementation of a semi-implicit discretization scheme originally applied to non-linear diffusion filtering [182]. Only drawback is the assumption that the slope of the *Level-Sets* guarantees $\|\nabla\phi\| = 1$ across iterations, with an evolving equation based on the formulation of [23]. This condition is not met with the *Level-Sets* formulation, but another model developed in [69] overcomes this drawback.

An interesting algorithm was derived in [137]: the *Hermes* algorithm is based on the idea to propagate the pixel that evolves faster at each step, thus combining the *Narrow-Band* and *Fast-Marching* approach, with propagation over a relatively small window. However, the proposed algorithm does not solve exactly the given partial differential equation since the evolution is computed locally, and this acceleration step is not valid in an homogeneous media, where the whole curve is propagating with the same speed (as shown for a pure advection flow in figure 4.6).

5.4.3 Combining two complementary methods

In the domain of medical image analysis, time-consuming algorithms are synonymous with non-interactive methods, and are therefore limited to a very small number of specific applications. In others words, the computation times of level sets are an obstacle to a wider use of these methods. The strategy we used is an original approach which tries to go beyond this classical barrier.

Both methods have a common advantage : they have no constraints or hypothesis on topology, which may change during convergence.

Our segmentation strategy will consist in using *Fast-Marching* method as a pre-segmentation tool, and then refine the segmentation with a level set method. This approach combines the advantages of both methods: the fast-marching pre-segmentation is rough but quick, and the level set needs only a few iterations to produce the final, highly accurate segmentation.

By combining fast-marching and level sets, we build a tool which is able to produce highly accurate segmentations of topologically and geometrically complex structures in minutes where level sets alone took hours. The framework is the following

Fast-Marching will provide a fast and rough initialization, near the solution. As a second process it will give a statistical study of the pre-segmentation (for the level-sets forces): looking at the local distribution of the different regions extracted with the competitive front, we are able to give the initial values of the region descriptors. Computing the mean and variance of different regions, we can skip the **MDL** statistical study and assume that those values are the original first and second moments of the Gaussian probabilities in section 5.2. And *Level-Sets* will start from the segmentation step achieved by *Fast-Marching* . In a few iterations, they will converge with more complex external forces, including region-based terms as in sections 4.4 and 5.2. Their particular formulation will lead to an accurate position of the sub-pixel iso-surface, that enhances visualization and measurements of pathologies.